超级计算面临的挑战
及其对未来数值算法设计的可能影响

田荣

摘要 超级计算机浮点运算能力的迅速提高和数据移动能力的增长缓慢已经造成了非常明显的"浮点效率鸿沟"。"浮点效率鸿沟"反映了传统数值算法与新兴硬件结构之间发展的不平衡和不匹配。在目前的新兴众核处理器上，一个"高效"的数值算法应该使单位访存所完成的浮点运算次数尽量加大，从而尽可能地享受由这一轮技术变革带来的新的"免费午餐"——超强浮点运算能力。这极可能将导致数值算法设计在思路上、甚至原则上的根本性转变。本文尝试面向新兴计算机体系结构，从充分释放众核处理器"冗余"计算能力的角度出发，发展一种新的高效且高精度（无额外自由度）的广义有限元方法，结合无网格/粒子/广义有限元法（以及有限差分）等一类新兴数值方法中所具有的"计算密集度可调"的共性特征，探讨硬件效能和数值精度"双赢"的新型计算模式的可能性。

关键词：百亿亿级计算 浮点效率鸿沟 众核 协同设计 广义有限元 无网格 粒子法

1 百亿亿次计算的挑战和超级计算机的"浮点效率鸿沟"

表1 目前的千万亿次系统与未来百亿亿次系统（原型设计）的比较.

比较内容	泰坦(Titan) (2011)	天河-II (2013)	百亿亿次系统 (美国 DOE 方案 2020-2022 部署)
峰值	27Pflop/s	54.9Pflop/s	1.2Eflop/s
功耗	8.3MW (2Gflop/W)	17.8MW (1.935Gflop/W)	~20MW (50Gflops/W)
系统内存	710TB ((32GB+6GB)×18688)	1.4PB （(64GB+3×8GB) ×16000)	32-64PB
节点峰值	1452Gflop/s (141+1311)	3431Gflop/s (422+3009)	1.2 or 15Tflop/s
节点内存带宽	232GB/s (52+180)	304GB/s(?) (64(?)+240)	2-4TB/s
节点并发度	16CPU+ 2688 Nvidia cores	24CPU+ 3×57Xeon Phi cores	***O***(10K) or ***O***(1K)
节点总网络带宽	8GB/s	6.35GB/s (MPI broadcast)	200-400GB/s
系统节点数	18688	16000	***O***(100K) or ***O***(1M)
系统总并发数	560640 (299008 AMD cores + 261632 Nvidia cores)	3120000 (384000 Ivy Bridge cores + 2736000 Xeon Phi cores)	***O***(1B)
平均无故障时间	?	?	O(<1day)

根据 2013 年 11 月的超级计算机 500 强（Top500）排名，中国的天河 II 号（CPU-MIC 异构架构）成为目前世界上最快的超级计算机。Linpack 实测速度为 54.9 petaflop/s（1 petaflops 为每秒 10^{15}（千万亿）次浮点运算）。第二名为安装在美国橡树岭国家实验室的泰坦（Titan，CPU-GPU 异构架构），Linpack 实测速度为 17.59 petaflop/s。目前，"百亿亿次"——比千万亿次快 1000 倍的——超级计算技术正在成为世界超算大国共同挑战的目标。

美国能源部按照电力消耗上限（如不应超过整个胡佛水坝的供电能力），以及日常运维、保有费用与系统造价的比例关系（如保有费用不应超过建造成本），确定出未来百亿亿次计算机设计目标，其中总体功耗不应超过 20MW[1]。表 1 列出了目前世界排名第一和第二的天河 II 号和泰坦(Titan)以及美国能源部的百亿亿次系统原型设计的主要指标。通过表 1 中的比较，我们可以发现两个事实：(1)天河 II 号的峰值为泰坦的 2 倍，功耗也正好为泰坦的 2 倍多一点；(2) 两台机器的功耗已分别达到 8.3MW 和 17.8MW，而峰值仅分别为百亿亿次的 2.7%和 5.49%。事实(1)说明：以天河 II 号和泰坦为代表的目前的技术进步模式是线性增量式的，性能与能源消耗成比例提高。事实(2)说明：在总体 20MW 的功耗约束下，目前的线性增量式的技术进步将无法实现百亿亿次计算目标，必须寄希望于计算技术的“颠覆性”进步。功耗效率成为了实现百亿亿级计算所面临的真正挑战[1]。

另一方面，在大规模科学与工程计算领域，人们熟知：一个并行有限元/有限差分程序，即使可获得接近线性的良好并行加速[2]，程序所能用到硬件的浮点性能也只占到峰值性能很少的一部分。无论是以单元计算为主的显格式计算（即无需解总体线性方程组，如差分模板（stencil）计算），还是以线性方程组求解（核心算法多为稀疏矩阵-向量乘（Sparse Matrix-Vector multiply，spMV））为主的隐格式计算，这个比例一般很少超过 30%。表 2 中，我们针对差分模板计算和稀疏矩阵-向量乘运算，对 AMD Opteron 6274 和 Intel Ivy Bridge 两款最新多核 CPU，以及英伟达 k20x GPU 和 Intel MIC 两款主流众核处理器的“理论浮点效率上限”进行了估算。不考虑时间局部性，即使是理论上的浮点效率(假设计算数据可以全部放入缓存)，结果仍然均在 31%以下。另外，根据最新 Top500 排名以 HPCG（采用预条件共轭梯度法求解线性方程组的有限差分计算）对现有

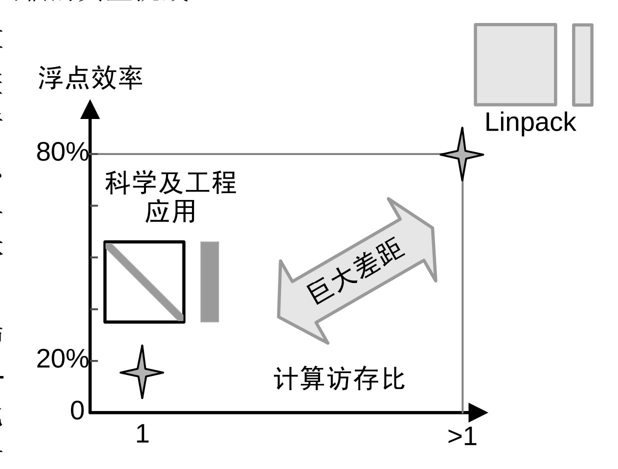


图1. 超级计算机的“浮点效率鸿沟”

表2 差分模板计算和迭代法求解线性方程组计算的理论浮点效率上限.

	AMD Opteron 6274	Intel Ivy Bridge	nVidia k20x GPU	Intel MIC
峰值(Gflop/s)	141	422	1311	3009
峰值带宽比(m)	21.69	52.75	58.26	100.3
3 点差分(3D)	6.5%	2.7%	2.4%	1.4%
5 点差分(3D)	14.4%	5.9%	5.3%	3.1%
7 点差分(3D)	22.3%	9.2%	8.3%	4.8%
9 点差分(3D)	30.2%	12.4%	11.2%	6.5%
spMV	9.22%	3.79%	3.43%	1.99%

超级计算机的测试结果表明，(即使相对 LINPACK 实测值)最好的浮点效率仅为 4.1%（日本的“京”超级计算机）（表 3）。然而，用测试标准程序 Linpack 测得的超级计算机的浮点效率一般可以达到 80%以上。日本的“京”超级计算机的 Linpack 测试结果甚至高达 93%。实际科学与工程应用的浮点效率与超级计算机的 Linpack“名义”浮点效率之间形成了一个明显的“浮点效率鸿沟”（表 3）。

[1] 从长远看，成功迈向百亿亿级计算的真正意义，绝不仅仅是完成一台百亿亿次计算机的建造，它将触发整个计算技术产业，包括从超级计算机到手持设备的颠覆性技术进步。比如，百亿亿级时代，你的手机处理能力将更强、待机时间却更长。

[2] 加速比为一个并行程序/并行算法的并行性能的主要评价指标。在问题规模一定的情况下，使用两个处理器核应该获得比一个处理器核快一倍的计算速度，即理论加速比应为 2；以此类推，对 n 个处理核，理论加速比应线性增加到 n。一个可获得线性加速的并行程序/算法即被认为是很好的并行程序/并行算法。

因此我们目前面对的现状是：一方面制造百亿亿次超级计算机面临越来越大的能耗挑战，另一方面，目前的超级计算机的实际浮点效率却非常低下。

表3 现有超级计算机在真实应用下的浮点计算效率(数据来源：Jack Dongarra)

安装场所	计算机	计算核数目	Linpack 峰值 (pflops)	Top500 排名	真实应用峰值 (pflops)	真实应用峰值 /Linpack 峰值
中国广州超算中心	天河-2 Xeon12c 2.2GHz +Intel Xron Phi 57c+定制核	3,120,000	33.9	1	0.580	1.7%
日本理化高级计算研究院	京 超级计算机 富士通 SPARC 64VIIIf×8C +定制核	705,024	10.5	4	0.427	4.1%
美国能源部橡树岭国家实验室	泰坦 Cray XK7 AMD16C+ Nvidia Kepler GPU 14C +定制核	540,640	17.6	2	0.322	1.8%
美国能源部阿贡国家实验室	Mira BlueGene/Q Power BQC 16C 1.60GHz +定制核	786,432	8.59	5	0.101	1.2%
瑞士国家超级计算中心	P1z Daint Cray XC30,Xeon 8C +Nvida Kepler 14C+定制核	115,984	6.27	6	0.099	1.6%

产生“浮点效率鸿沟”的原因并不复杂：科学与工程计算相当一部分为求解线性方程组。目前大型线性方程组求解主要采用克雷洛夫（Krylov）子空间迭代法[3]，这类算法最终的核心多归结为稀疏矩阵-向量乘，而稀疏矩阵-向量乘算法的“计算访存比”不超过2：1（单位：flop/word，word为浮点字长，为4或8字节），是典型的访存密集型运算(通常人们笼统地称科学与工程计算为计算密集型，是与其它的计算机应用类型相比较而言的)。而Linpack标准测试所对应的算法是稠密矩阵的直接法求解，算法核心为双精度矩阵相乘（Double precision GEneral Matrix Multiply（DGEMM）），“计算访存比”为 n：1（n 为分块矩阵大小），属于计算密集型运算。显然，两类算法的“计算访存比”具有本质性差别。而正是算法上的这种本质性的差异直接决定了Linpack测试和实际科学与工程应用之间在浮点效率上的明显差距。

而且坏消息是，这种差距目前还在进一步扩大。

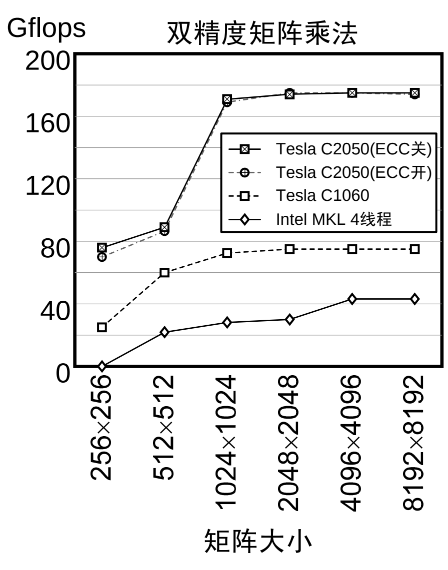
2 众核计算——进一步扩大的“浮点效率鸿沟”

图2数据源自英伟达公司[2]，用于显示众核GPU相对于多核CPU的性能优势。基于同样的数据，我们对Tesla C1060和Tesla C2050两款众核GPU处理器做一个横向比较。C2050的双精度浮点峰值速度是C1060的6.6倍。如果我们问：“对于科学与工程计算，C1060和C2050哪一款产品更好？”PC时代的经验是峰值速度越高越好。

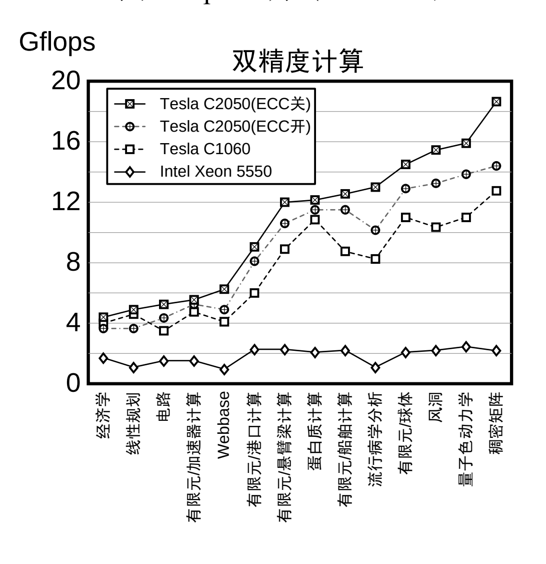
图2中左图为两款GPU在进行“双精度稠密矩阵-矩阵乘”运算时的性能差异。计算通过调用cuBLAS3.1库中的DGEMM子函数实现。DGEMM正是超级计算机性能评测程序Linpack的核心算法。图2中右图则是针对13个真实科学与工程应用程序和一个稠密矩阵计算程序——其中大多数为有限元/有限差分计算——的测试结果。这些真实应用的核心算法以稀疏矩阵-向量乘为主。我们看到，对于Linpack测试而言，C2050速度是C1060的两倍，

[3] 典型代表有共轭梯度向量法CG，广义最小残量法GMRES，和双共轭梯度法BiCG等。

而对于真实应用，C2050 仅仅比 C1060 快了约 40%——对于 Linpack 测试，C2050 和 C1060



(a) 双精度稠密矩阵乘法 DGEMM



(b) 双精度稀疏矩阵-向量乘

图2. Tesla C2050 和 Tesla C1060 两款众核 GPU 处理器的性能测试

原始数据来源：英伟达公司网站[4]。对于 Linpack 测试：C2050 比 C1060 性能提升显著；对于实际的科学与工程计算：前者无特别明显的"能效"优势。

性能差异明显，而对于真实应用，二者性能则非常接近。

表4 C2050 和 C1060 GPU 的主要性能指标

GPU	功耗	峰值	内存带宽
C1060	188W	78Gflop/s	102GB/s
C2050	247W	515Gflop/s	144GB/s

造成这种性能表现上的差异的原因如下：DGEMM 是计算密集型操作，其性能提升理论上由硬件的浮点计算能力的提升决定（但实际上，即使这一计算密集型算法，在 C2050 上也受到 GPU 寄存器大小的限制，使得分块矩阵 n 不能很大，大大降低了实际计算的计算访存比）。而稀疏矩阵-向量乘是典型的访存密集型操作，其性能提升主要由硬件的访存带宽的提升幅度决定。C2050 的访存带宽（144GB/s）比 C1060（102GB/s）高出 46%。这正好解释了为什么对于真实应用，C2050 仅仅比 C1060 快了 40%。实际上，如果考虑功耗、性价比，特别是功耗因素，C1060 与 C2050 相比，对于科学与工程计算而言可能并不算差。如果从浮点效率（能耗效率）来看，C1060（14%）反而比 C2050（3%）更高——虽然 C2050 比 C1060 浮点性能快 6.6 倍，但效率相比 C1060 却大大降低。

图 3 进一步综合性地给出了典型应用算法在目前四款主流处理器上的浮点效率的变化趋势。我们发现，由于真实科学与工程计算的访存密集的特征所限，它们并没有简单地从浮点性能的单一增长中获益，反而出现浮点速度越快，浮点效率越低的怪现象。

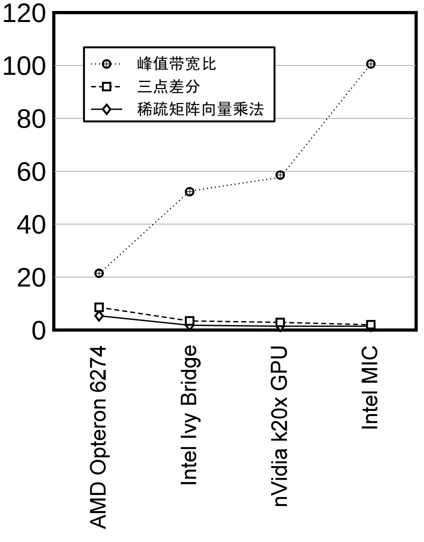


图3. 处理器性能与典型应用的浮点效率变化曲线

由于真实科学与工程计算的访存密集的特征所限，它们并没有简单地从浮点性能的单一增长中获益，反而出现浮点速度越快，浮点效率越低的怪现象。

浮点效率低下即计算能力的浪费。在"功耗"成为百亿亿次超级计算机的"首要物理约束"

的今天，缩小“浮点效率鸿沟”具有极其重要的现实意义。

3 难点所在：传统数值算法与新兴众核硬件结构之间的不匹配

“浮点效率鸿沟”的存在，不是计算机硬件设计不好。在过去 40 年中，CPU 核心的速度提高了近千倍，但是内存的读写速度提高却不大；更为关键的是，内存的延迟没有降低。如从 DDR 到 DDR3，内存的传输速度成倍地提高，但是内存延迟[4]没有减少，反而还有所增加。这使得内存延迟已经成为了制约 CPU 实际处理能力提高的瓶颈。在实际应用中，CPU 始终都在等待内存访问。当内存端口满负载时，CPU 核心有超过 50%时间是空闲的。这就是早在 1994 年便被提出的“内存墙(Memory Wall)”[3]问题。目前，非易失性存储器（NVM）以及三维封装技术（3D Integration）可望成为具有“颠覆性”的新一代存储技术。但是无论采取何种快速存储技术，计算机的冯·诺伊曼“存储程序原理”架构可能是造成内存墙问题最根本的原因。克服内存墙将是一个长期、巨大的技术挑战[4]。

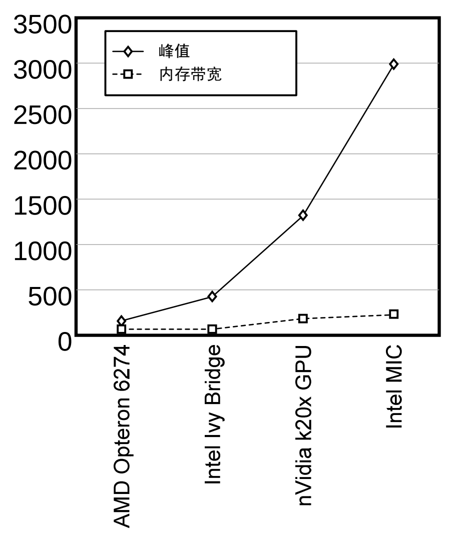


图4. 性能与访存带宽的发展趋势

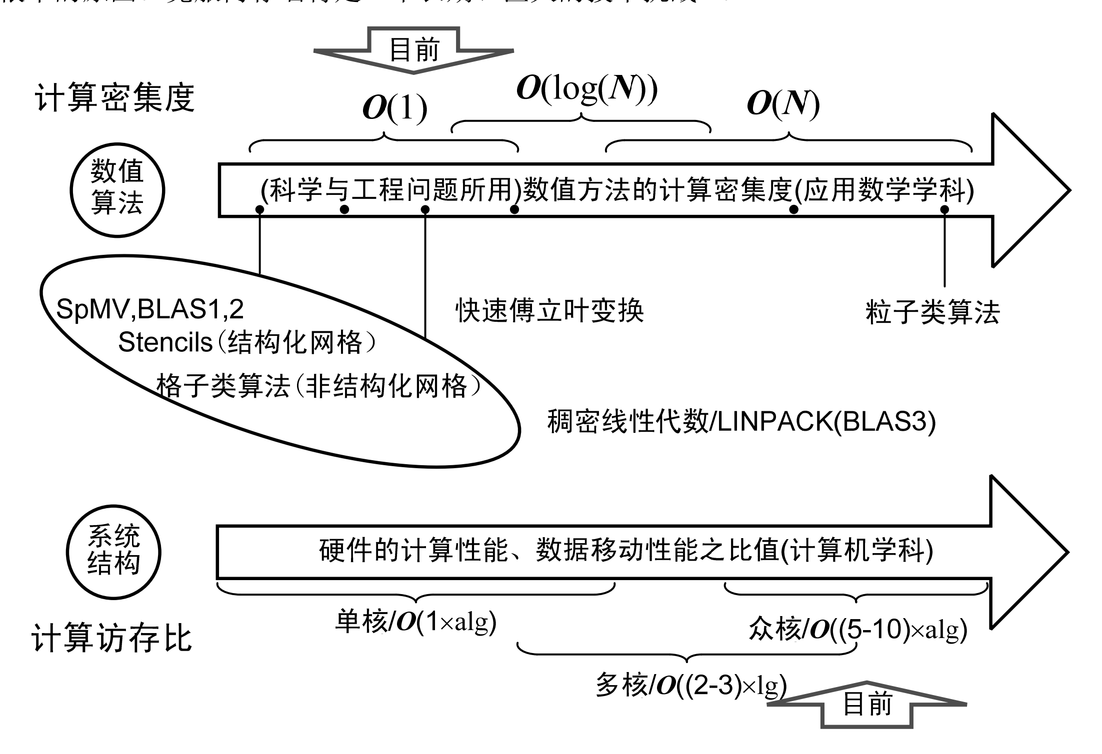


图5. 处理器结构与数值算法在基本计算特征上的不匹配和不协调
“alg”为浮点格式字长，单精度为 4，双精度为 8

“内存墙”问题和新兴多核/众核技术的发展正在使得处理器计算能力和访存能力之间的差距愈加扩大[5]。如图 4 所示，我们同样针对 AMD Opteron6274 和 Intel Ivy Bridge 两款最新多核 CPU，以及英伟达 k20x GPU 和 Intel MIC 两款最新众核处理器，画出峰值与访存带宽增长趋势。从中可以看出，处理器性能呈现指数级增长，而访存能力几乎呈线性增长或微弱

4 内存延迟是指内存在接到访问命令后，要等一段时间，才能传回数据。这个延迟超过 100 纳秒。过去 30 年，DRAM 内存的传输速度（仅仅是传输数据的速度）有所提高，但是 DRAM 的延迟不但没有什么改善，反而有继续扩大的趋势[5]。

的指数级增长，二者间的差距也呈指数级增长。实际上，从单核、多核到众核，处理器的“计算访存能力比值”越来越大，已经发生了约一个数量级的改变（见图 5 中计算访存比的标注）。

“计算访存能力比值”高意味着处理器更加喜欢高“计算密集度”的数值算法，即单位访存所进行的浮点运算次数很多的数值算法。然而，数值算法或应用数学学科的发展却是非常缓慢的——与计算机处理器技术激进演化的发展节奏完全不同。从图 5 可以看出，如果按照数值方法的计算密集度进行归类，我们会发现：目前 80-90%的科学与工程计算的核心算法仍然以“线性计算密集度”算法为多，密集地集中于箭头的左端。而硬件的计算访存能力比值却远远位于箭头的右端。处理器技术的激进快速发展与相对发展缓慢的数值算法之间已经出现明显的不适应和不协调。

4 研究现状及发展趋势——“好”的数值算法的定义在发生改变

综合上述表 1、图 3 和图 4，我们可以归纳出以下基本事实：

（1） 典型真实应用的浮点效率远远小于 Linpack 效率（对于以稀疏矩阵-向量乘为核心算法的线性方程组求解，效率均低于 10%）。而且，峰值带宽比越高，实际应用的浮点效率反而越低。

（2） 稀疏矩阵-向量乘（线性方程组克雷洛夫子空间迭代法求解的核心计算）“计算访存比”固定，为最典型的访存密集型计算，浮点效率最低。

（3） 理论上，差分模板计算的效率可以通过改变差分格式得到改善。

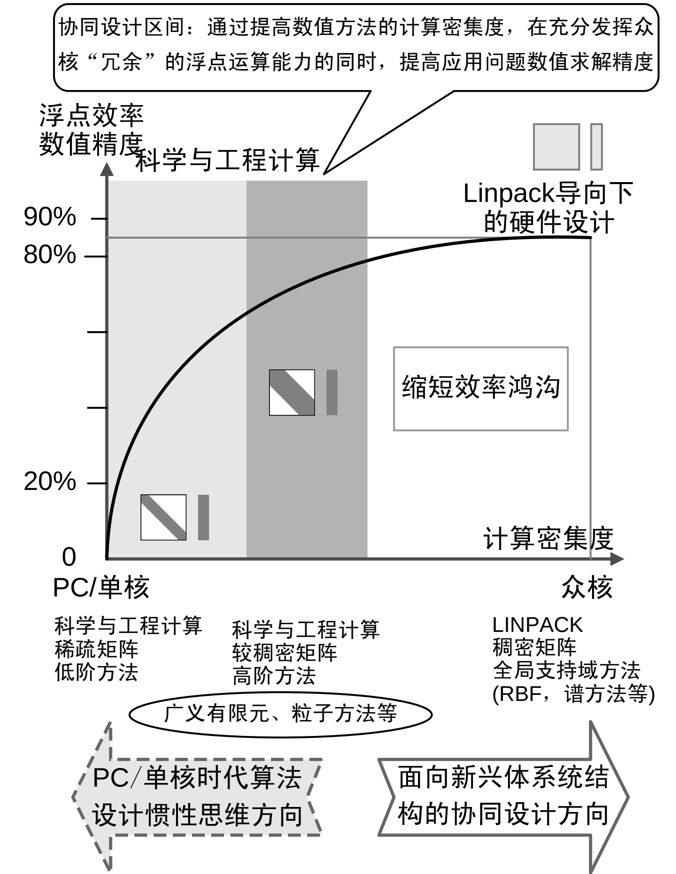


图6. 面向新兴处理器体系结构的数值算法的协同设计

事实(1)说明，处理器的峰值或峰值带宽比越高，实际应用的浮点效率越低，反映了正在加剧的“浮点效率鸿沟”问题，反映了计算机硬件与数值算法之间的不适应和不协调。事实(2)表明，需要针对新兴计算机体系结构，在算法层面对线性方程组的求解方法进行改变或重新选优，以避免过低的浮点效率——在目前的多核/众核处理器上，如果关键数值算法不作出改变，浮点效率将无法得到真正有效的改善。事实(3)揭示了克服面向浮点效率问题时，我们可以采取的具体措施——改变数值算法。通过改变差分模板的阶次(低阶$\rightarrow$高阶)，在差分阶次或差分精度提高的同时，浮点效率也得到了改善（计算效率和算法精度的双赢）。这说明“好”算法的定义正在发生改变。

过去被认为是计算“费力”的高精度算法可能是未来的“好”算法。或者简单地说，在未来百亿亿次系统上，科学与工程应用算法应该使单位访存所完成的浮点运算次数最大化，从而

尽可能地享受新的"免费午餐"——浮点计算能力。在数值方法设计方面，在 PC 时代，由于浮点运算能力受限，算法设计总是喜欢"线性计算密集度"的算法，喜欢稀疏性。如图 6 所示，过去我们沿"向左"箭头方向思考。而面对众核的高浮点运算能力和数据移动能力的相对不足，未来应该强调矩阵"适当"稠密，增加计算密集度，鼓励沿"向右"箭头方向思考。这将带来科学与工程应用算法设计在思路上甚至原则上的根本性转变。

5 面向众核计算的数值方法协同设计

按照图 6 的预测：无网格、粒子类方法以及（单位分解）广义有限元等一类过去被认为计算"费力"的但具有更好的数据局部性的新兴数值方法值得我们从发挥众核计算能力的角度重新给予关注。

传统上，数值方法研究关注的只是精度、收敛性等数学指标，从来都不会去关注"计算密集度"这一来自于计算机新兴体系结构对数值算法的新要求。受到表 2 中有限差分浮点效率变化的启发，图 7 对无网格/粒子类方法进行了类似的考查。假设胞元（cell）中的平均粒子数为 $\boldsymbol{n}$。中心胞元内粒子的重用近似等于（或小于）周围胞元内粒子总和，为 $\boldsymbol{n}(3^{\boldsymbol{d}-1})$，$\boldsymbol{d}$ 为问题维数。周围胞元内粒子的重用大致等于（或小于）中心胞元内粒子个数，为 $\boldsymbol{n}$。这一类方法的"平均"计算密集度为 （$2\boldsymbol{n}(3^{\boldsymbol{d}-1})$）/$3^{\boldsymbol{d}}$。

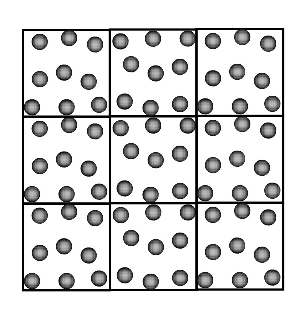


图7. 无网格/粒子类方法

表5 无网格/粒子类方法"单元"计算的理论浮点效率

Cell 中粒子数	AMD Opteron 6274	Intel Ivy Bridge	nVidia k20x GPU	Intel MIC
2×2×2(*n*=8)	71.0%	29.2%	26.4%	15.3%
3×3×3(*n*=27)	100.0%	98.5%	89.2%	51.8%

对这一类数值算法，我们可以针对前面列出的四款处理器估算出理论浮点效率。如表 5 所示，我们发现，在同样处理器上，通过提高算法精度，浮点效率也大幅提升。理论分析表明，这类无网格/粒子类方法以及前面提及的有限差分方法，一个最大的共性特点是，通过改变算法格式（差分阶或胞元/影响半径的大小）可以主动控制处理器的浮点效率。

受此启发，我们基于前期发展的无额外自由度的广义有限元方法[46][47][48]，探讨在众核计算平台上计算密集度调节的重要性和意义。下面先简单介绍这一新的广义有限元方法。

无额外自由度的广义有限元方法

单位分解（partition of unity）广义有限元方法的研究一般认为源于梅伦克（Melenk）和巴布什卡（Babuška）的单位分解方法[20]和单位分解有限元方法[21]（类似的思想又可追溯到该作者早期的工作[22]）。几乎同时，杜阿尔特（Duarte）和奥登（Oden）[23][24]提出类似的方法，称为 *hp*-cloud 方法。在思想上非常相近但发表时间更早的工作，可见于由我国旅美学者石根华提出的数值流形方法[5][39]。单位分解函数概念本身非常简单，既可使用无网格插值方法构造，也可直接使用有限元的形函数。作为有限元方法的自然延伸，基于有限元形函数的单位分解方法获得了极大的关注和成功的应用，如斯特劳鲍里斯（Strouboulis）和巴布什卡[30]-[37]和杜阿尔特等深入发展的广义有限元[25]-[29]。为了清晰指代，下面的叙述中"广义有限元法(GFEM)"主要指代基于有限元形函数的单位分解方法。

[5] Numerical Manifold Method，NMM

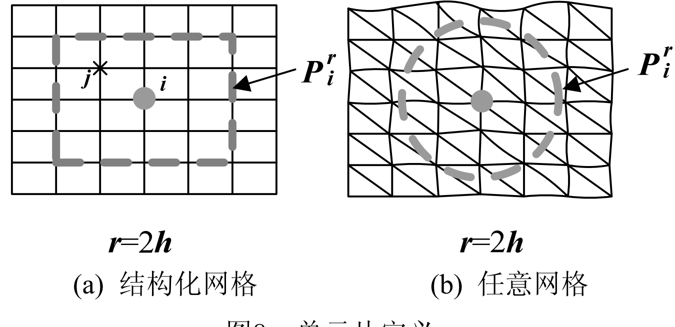
广义有限元的核心是单位分解逼近：

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{i\in I} \boldsymbol{N}_i \boldsymbol{u}_i + \sum_{i\in I} \boldsymbol{N}_i \sum_k \varphi_k^i \boldsymbol{a}_{i(k)} \tag{1}$$

其中 $\sum_i \boldsymbol{N}_i(\boldsymbol{x}) \equiv 1$ 构成了全域上的一个单位分解，φ_k^i 为节点 $\boldsymbol{i}$ 支撑域上的用户自定义局部近似函数，$\boldsymbol{a}_{i(1)}$，$\boldsymbol{a}_{i(2)}$，…为局部近似函数引入的节点 $\boldsymbol{i}$ 的广义自由度或额外自由度。

在现有的广义有限元方法中，节点上自由度个数会随着局部近似函数的阶次变化而变化，因此提高局部函数的阶次不会改变该方法的计算密集度（见稍后图 12 的说明）。下面，我们提出一种“无广义自由度的广义有限元方法”（下称“新方法”）。

假设 $\boldsymbol{P}_i^r$ 为环绕节点 $\boldsymbol{i}$ 的所有单元组成的“单元片（nodal patch）”，$\boldsymbol{r}$ 代表单元片尺寸。根据网格特征，单元片尺寸可通过两种方式定义：对于规则结构化网格或非均匀网格，为环绕节点 $\boldsymbol{i}$ 的 $\boldsymbol{m}\geq 1$ 层单元的支撑域的大小（见图 8(a)）；对于任意均匀网格，则可简单取为节点 $\boldsymbol{i}$ 的影响圆（或球）半径（见图 8(b)）。节点 $\boldsymbol{i}$ 定义为“片星（patch star）”。若无特别说明，指标 $\boldsymbol{i}$ 做特意保留，特指“片星节点(patch star)”，指标 $\boldsymbol{j}$ （$\boldsymbol{j}\neq\boldsymbol{i}$） 表示“非片星节点（non-patch star）”。$\{\boldsymbol{x}_k | \boldsymbol{x}_k \in \boldsymbol{p}_i^r\}$ 表示单元片 $\boldsymbol{P}_i^r$ 上所有节点的集合。



(a) 结构化网格　　(b) 任意网格

图8. 单元片定义

首先，使用节点 $\{\boldsymbol{x}_k | \boldsymbol{x}_k \in \boldsymbol{p}_i^r\}$ 构造 $\boldsymbol{u}_i(\boldsymbol{x})$，$\boldsymbol{x}_k \in \boldsymbol{p}_i^r$ 在单元片上的一个局部近似：

$$\boldsymbol{u}_i(\boldsymbol{x}) = \sum_{k=1}^{n_i} \varphi_k^i(\boldsymbol{x}) \boldsymbol{u}_k \tag{2}$$

其中 $\varphi_k^i(\boldsymbol{x})$ 为局部函数，$\boldsymbol{n}_i$ 为单元片上的节点数，$\boldsymbol{u}_k$ 为节点未知数。

将局部近似 $\boldsymbol{u}_i(\boldsymbol{x})$ 直接取代标准有限元的节点位移 $\boldsymbol{u}_i$

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{N}_i(\boldsymbol{x}) \boldsymbol{u}_i \tag{3}$$

则可得出一种新的逼近形式

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{i=1}^{N} \left(\boldsymbol{N}_i \left(\sum_{k=1}^{n_i} \varphi_k^i \boldsymbol{u}_k \right) \right) \tag{4}$$

插值(4)与插值(3)均基于同一有限元网格的节点构造，并没有引入新的节点或自由度。将式(4)中的项展开、然后按相同节点归类、重新组合后可得到

$$u^h(x) = \sum_i \left(N_i \varphi_i^i + \sum_{k\in J_i, k\neq i} N_k \varphi_i^k \right) u_i \tag{5}$$

其中 φ_i^k 为在单元片 $\boldsymbol{k}$ 上节点 $\boldsymbol{i}$ 的局部函数(注意 φ_i^k 和 φ_k^i 并不等价)，$\boldsymbol{n}_i$ 为单元片 $\boldsymbol{P}_i^r$ 上的节点数，或包含节点 $\boldsymbol{i}$ 的所有单元片的数量，$\tilde{\boldsymbol{N}}_i$ 表示新的形函数。

比较(4)和(1)，可以清楚地看出，(4)实质上也是一种单位分解逼近。然而，新单位分解逼近具有两个独一无二的特点：(a)不包含任何广义自由度；(b)只要局部逼近函数在各自的

“片星”处插值，则无论其在“非片星”处插值与否，最终构造出的逼近函数一定是全局插值的。

新方法的特性逐条详细讨论如下：

（1） 具有现有广义有限元方法的高阶性质。通过增大单元片尺寸，可以得到“高阶”的局部函数，从而实现高阶插值。

（2） 不包含广义自由度，并且待求的总自由度不会随着局部函数的阶次改变而改变。这是现有广义有限元方法不具有的特点，这直接导致线性无关性（linear independence）和与标准有限元一样的稳定性[46]。这两点特性是与现有广义有限元方法最大、最本质的区别。

（3） 具有插值性质。局部函数可以采用某种插值方法，如拉格朗日插值或径向基函数等，或者采用具有“单点插值”特性的某种逼近方法来构造。对于后者，可以使用移动最小二乘法来构造局部函数，只需强制该最小二乘拟合函数在“片星处”插值即可——该“单点插值”约束很容易满足。在这些情况下，所构造出的全局逼近函数自然具有全局插值性质。

5.1 局部函数的构造——“单点插值”最小二乘逼近

局部函数的构造是基于单元片上的节点集来实现的。现有的标准化技术，如无网格方法[40][44][45]或者网格类方法，都可借用过来。唯一的要求是：如果使用逼近格式，比如移动最小二乘[41]，它应当而且只需在单元片的“片星”处具有插值性质。对于结构化网格，可以使用拉格朗日多项式插值；对于任意网格，可以使用径向基函数[42][43]构造局部函数。这些构造过程直观简单，这里不再赘述。

下面重点介绍我们提出的适用于任意网格的“单点插值”移动最小二乘逼近[46]-[48]。这一方法使我们能够使用局部逼近来构造出全局插值的广义有限元方法。我们仍然用指标 $\boldsymbol{i}$ 特指“片星节点”，$\boldsymbol{j}$ 表示“非片星节点”。在以 $\boldsymbol{i}$ 为中心的单元片上，局部逼近定义为

$$\boldsymbol{u}_i\left(\boldsymbol{x}\right)=\mathbf{p}^{\mathrm{T}}\left(\boldsymbol{x}\right)\boldsymbol{a}\left(\boldsymbol{x}\right),\boldsymbol{x}\in\boldsymbol{P}_i^r \tag{6}$$

其中基函数 $\mathbf{p}$ 可以包含解的先验知识，对于光滑问题，可以直接取为任意高阶多项式；对于具有局部特征的问题，如裂纹尖端，可以是非多项式特殊函数—简言之，基函数是可以根据待求问题解的先验知识进行自定义的，$\mathbf{p}$ 的长度与单元片尺寸 $\boldsymbol{ch}$ 相适应（$\boldsymbol{h}$ 为网格尺寸），$\mathbf{p}$ 也可以包含解的先验知识，如扩展有限元法[6]中使用的裂纹尖端基本解等，$\boldsymbol{a}$ 为待定系数。

为了构造一个在“片星”处插值的移动最小二乘逼近，我们使用下面的带有约束条件的最小二乘逼近

$$\boldsymbol{J}=\frac{1}{2}\sum_{\boldsymbol{k}=1}^{\boldsymbol{n}_i}\boldsymbol{w}_{\boldsymbol{k}}\left(\boldsymbol{x}\right)\left(\mathbf{p}^{\mathrm{T}}\left(\boldsymbol{x}_{\boldsymbol{k}}\right)\boldsymbol{a}-\boldsymbol{u}_{\boldsymbol{k}}\right)^2+\lambda\left(\mathbf{p}^{\mathrm{T}}\left(\boldsymbol{x}_i\right)\boldsymbol{a}-\boldsymbol{u}_i\right) \tag{7}$$

其中 $\boldsymbol{n}_i$ 是单元片 $\boldsymbol{i}$ 上的节点总数。为了便于程序实现，“片星节点”$\boldsymbol{i}$ 也包含在求和项中，这样式(7)的第一部分与标准移动最小二乘[41]完全一样，λ为拉格朗日乘子，用于强制 $\boldsymbol{u}_i(\boldsymbol{x})$ 精确满足“单点插值”条件

$$\mathbf{p}^{\mathrm{T}}\left(\boldsymbol{x}_i\right)\boldsymbol{a}=\boldsymbol{u}_i \tag{8}$$

对最小二乘范数取极值可得到“单点插值”移动最小二乘逼近

[6] Extended Finite Element Method , XFEM

$$\boldsymbol{u}_i(\boldsymbol{x}) = \sum_{k=1}^{n_i} \phi_k^{P_i^r}(\boldsymbol{x})\boldsymbol{u}_k \tag{9}$$

其中

$$\phi_k^{P_i^r}(\mathbf{x}) = \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x})\left(\mathbf{A}^{-1}\boldsymbol{p}_k - \frac{1}{\mathbf{A}_{11}^{-1}}\mathbf{A}_{(1)}^{-1}\mathbf{A}_{(1)}^{-\mathrm{T}}\boldsymbol{p}_k + \frac{1}{\mathbf{A}_{11}^{-1}}\mathbf{A}_{(1)}^{-1}\boldsymbol{\delta}_{ik}\right) \tag{10}$$

$$\mathbf{A} = \sum_{k=1}^{n_i} w_k \mathbf{p}_k \mathbf{p}_k^{\mathrm{T}} \tag{11}$$

$$\mathbf{p}^{\mathrm{T}} = \left[1, \frac{\boldsymbol{x}-\boldsymbol{x}_i}{\boldsymbol{ch}}, \frac{\boldsymbol{y}-\boldsymbol{y}_i}{\boldsymbol{ch}}, \ldots\right] \tag{12}$$

$\mathbf{A}_{(1)}^{-1}$ 为 $\mathbf{A}^{-1}$ 的第一列，$\mathbf{A}_{11}^{-1}$ 为 $\mathbf{A}_{(1)}^{-1}$ 的第一个元素，δ 为克罗内科函数（Kronecker delta）。可以证明，当 $\boldsymbol{x} = \boldsymbol{x}_i$ 时，将 $\mathbf{p}_i^{\mathrm{T}} = [1,0,0,\ldots]$ 带入可得出 $\phi_i^{P_i^r}(\boldsymbol{x}_i) = 1$，从而说明局部函数在“片星节点”处满足插值性质。

在公式(7)中，令 $\boldsymbol{w}_k$=1，“单点插值”移动最小二乘便退化为“单点插值”最小二乘（SILS）。相对“单点插值”移动最小二乘，“单点插值”最小二乘（特别是其导数）具有计算速度快的优点。

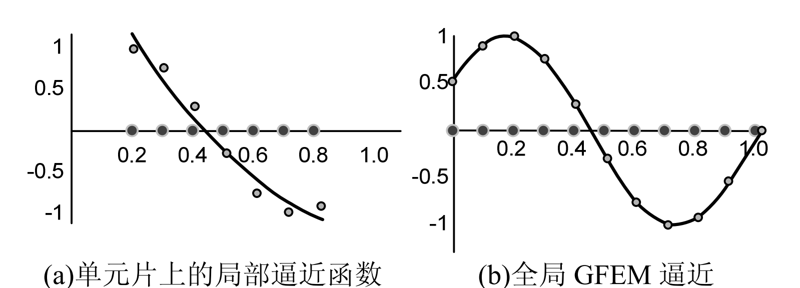
使用 SIMLS / SILS 作为局部逼近，新广义有限元法可以构造如下

$$\boldsymbol{u}^h(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{N}_i\left(\sum_{k=1}^{n_i} \phi_k^{P_i^r}\boldsymbol{u}_k\right) \tag{13}$$

其中 $\phi_k^{P_i^r}$ 为单元片 P_i^r 上的 SIMLS 或 SILS 局部函数。

SIMLS/SILS 适用于任何维数的规则或不规则网格。基于 SIMLS/SILS 的局部函数和相应的新的广义有限元插值对正弦函数的拟合结果示于图 9。从图中看到，虽然局部函数本质上是逼近的，但导出的广义有限元却是全局插值的。

小结：移动最小二乘本来只是一种拟合逼近，但由此导出的新广义有限元却是全局插值的，严格满足克罗内科函数性质。同时，移动最小二乘具有性能强健（无坏条件数问题）和计算开销小（如 SILS）等优点。



(a)单元片上的局部逼近函数　　(b)全局 GFEM 逼近

图9. 使用 SIMLS 局部逼近的广义有限元（GFEM）
r=3h，被逼近的函数为正弦函数

关于该方法更详细的内容可参考文献[46]。本文主要目的是以该方法为例，抛砖引玉，讨论过去被认为代价昂贵的一类算法在新兴众核处理器上可能出现的新的特征。

5.2 数值试验

5.2.1 收敛性测试

考虑下列的一维测试问题

$$
\begin{aligned}
&-\Delta \boldsymbol{u} = \boldsymbol{f} \quad \boldsymbol{x} \in (0,1) \\
&\boldsymbol{u}(0) = 0 \\
&\nabla \boldsymbol{u}(1) = 0
\end{aligned} \tag{14}
$$

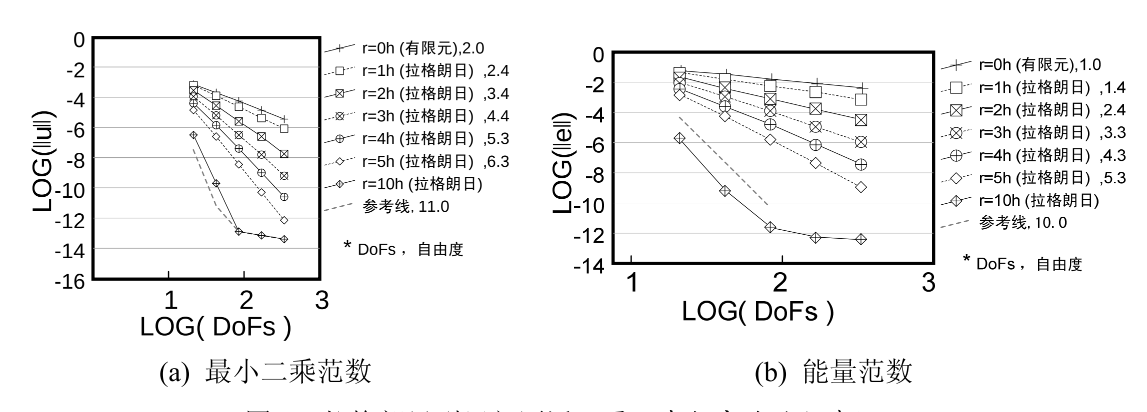
精确解取为

$$
\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{x}(\boldsymbol{x}-1)^{15} e^{-x^2} \tag{15}
$$

误差的最小二乘范数和能量范数分别定义如下

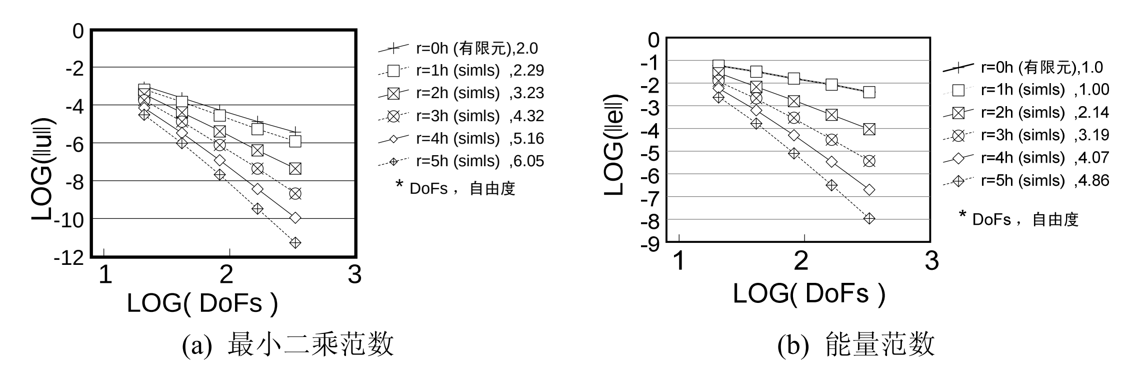
$$
\|\boldsymbol{u}\| = \left(\int_\Omega \left(\boldsymbol{u}^h(\boldsymbol{x}) - \boldsymbol{u}(\boldsymbol{x})\right)^2 \mathrm{d}\Omega\right)^{\frac{1}{2}}, \|\boldsymbol{e}\| = \left(\int_\Omega \left(\nabla \boldsymbol{u}^h(\boldsymbol{x}) - \nabla \boldsymbol{u}(\boldsymbol{x})\right)^2 \mathrm{d}\Omega\right)^{\frac{1}{2}} \tag{16}
$$

收敛测试中使用了五个一致加密的规则网格，分别为 20、40、80、160 以及 320 个线性有限单元网格。单元片大小取 $\boldsymbol{r}=\boldsymbol{ch}$，$\boldsymbol{c}$=0…，10。“$\boldsymbol{r} = 0\boldsymbol{h}$”对应于标准有限元情形。



(a) 最小二乘范数 (b) 能量范数

图10. 拉格朗日型局部逼近（图例中数字为收敛率）



(a) 最小二乘范数 (b) 能量范数

图11. SIMLS 型局部逼近（图例中数字为收敛率）

单元上高斯积分点数取为 $\boldsymbol{c}$+2。收敛性测试结果见图 10 和图 11，分别对应于两种不同的局部逼近构造方式。

图 10 和图 11 中的结果显示，单元片大小每增大 1$\boldsymbol{h}$，新广义有限元的收敛阶通常都会相应提高一阶（收敛速率见图例）。这一收敛属性类似于现有广义有限元，不同点在于：新广义有限元在网格尺寸和自由度数不改变的情况下，便可以获得不同阶的精度和收敛性。这种特征过去多见于有限差分方法。在传统的有限元或现有广义有限元中，要提高收敛阶，必须在相同网格尺寸下增加自由度，或在单元上增加新的节点（传统有限元），或增加节点上的广义自由度（现有广义有限元）。

5.2.2 新方法在新兴众核处理器 GPU 上的性能测试

由于总体方程求解与局部逼近方式无关，下面我们主要对新方法形成的总体方程的求解进行测试。对于单元的计算与组装，由于其与局部逼近方式有关，暂不考虑（但不影响结果的一般性）。

首先考察新方法与现有广义有限元方法在总体刚度矩阵形式上的差别。从图 12 中对新方法与现有方法总体刚度矩阵特征的比较，我们观察到：

- 在新方法中，总体矩阵的大小不随局部逼近阶次的改变而改变，而在现有方法中，总体矩阵的维数会随着局部逼近阶次的提高而增大。
- 在新方法中，总体矩阵的稀疏性（非零元素占比）随着局部逼近阶次的提高而降低，而在现有方法中，改变局部逼近的阶次无法改变总体矩阵的稀疏性。

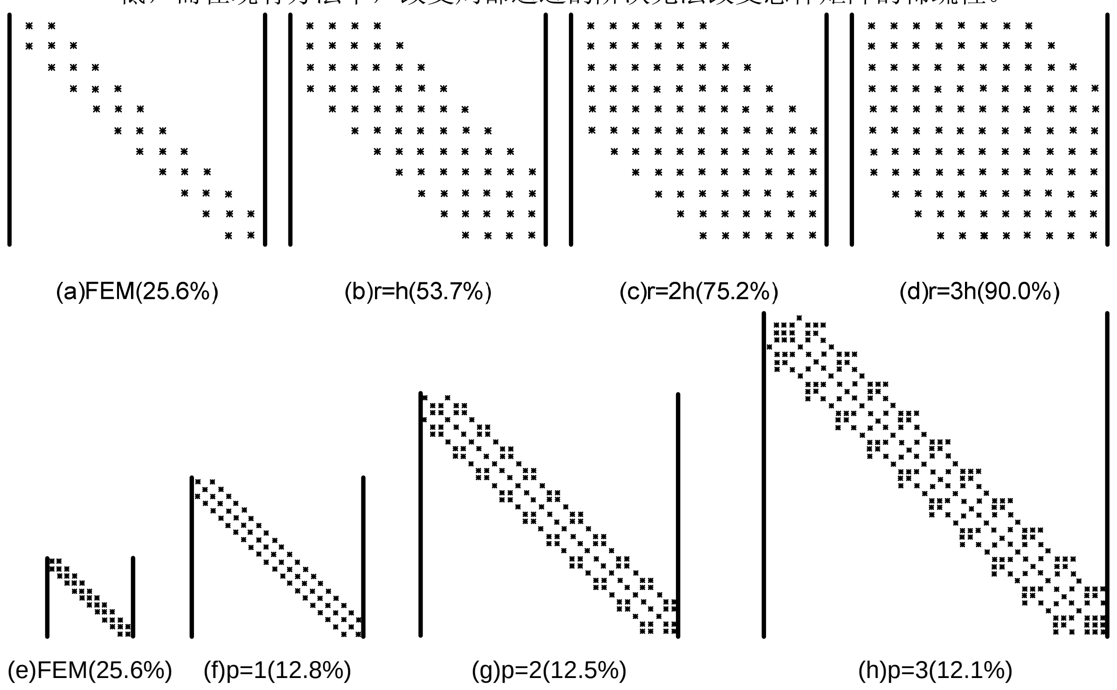


图12. 一维问题 10 单元网格对应的总体刚度矩阵

(a)-(d) 为新方法, (e)-(h) 为现有方法. 括弧中的数字为非零元素占比, 反映矩阵的稀疏性. 现有方法的总体矩阵随局部逼近的阶次的提高而几乎不变或变得更加稀疏, 而在新方法中, 随着局部逼近阶次的提高, 总体矩阵变得稠密.

在新方法中，总体矩阵的稀疏性是变化的或可调节的。这也是目前无网格/粒子类方法的一个共同特征。因此，下面的结论对于无网格/粒子类方法具有一定的普遍意义。

测试问题为二维泊松问题。测试平台为 GPU 众核和 CPU 单核（只用一个核进行计算以反映 PC 时代的计算特征）。性能指标为单位时间内浮点运算次数 Gflop/s。测试对象为总体方程组的求解环节（使得测试更具有一般性）。求解方法采用共轭梯度向量法。求解器采用基于 CUDA 的开源线性代数库 CUSP。稀疏矩阵采用 CSR 存储格式。测试结果示于图 13。

首先，令 $\boldsymbol{r}=0\boldsymbol{h}$ 将新方法退化为标准有限元方法，对标准有限元方法在 CPU 单核（PC 时代）与 GPU 众核上的性能进行比较，结果见图 13 中圆圈中的两个数据点。从图中我们观察到，虽然 CPU 单核与 GPU 众核的峰值性能相差很大（前者为 10Gflop/s，后者为 78Gflop/s），但标准有限元在两种计算平台上的性能却相差无几。这一测试结果表明：如果在众核上使用标准有限元算法，众核强大的计算能力将可能得不到充分发挥和利用（算法计算密集度低，性能受访存限制）。

然后，我们逐渐增加局部逼近的阶次（提高单元片大小 ***r***），对新方法的"高阶"格式在 CPU 单核和 GPU 众核上重新进行性能比较（图中实线）。我们发现，在 GPU 众核上，当提高方法的阶次，即增大单元片大小 ***r***（或降低总体矩阵的稀疏性），新方法的性能得到非常明显的提升（硬件性能得到了利用和发挥），与方法阶次几乎呈线性增长。这一测试结果表明：新方法的高阶格式能充分发挥众核的计算能力。然而，图 12 显示新方法导致较为稠密的总体矩阵，如果从 PC 时代的传统观点看，这类方法应该是较为低效的算法。上述测试却表明，在众核上由于硬件性能得到了充分利用，这类方法反而变得更加高效。

图 13 中虚线表示同样的高价方法在 CPU 单核上的测试结果。我们看到，如果在 CPU 单核上（或 PC 时代），高阶格式并不能提高硬件的计算性能利用率，反而会导致整体计算性能下降。这一结果也与在 PC 时代，高阶方法或那些导致稠密矩阵的方法被认为是低效或计算费力的事实是相符的（在 PC/单核时代，硬件的计算能力是主要瓶颈）。

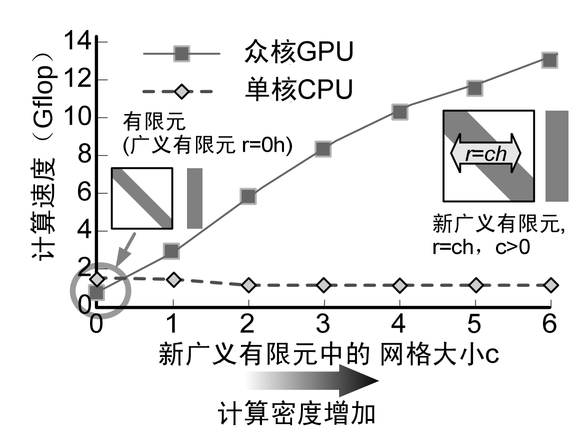


图13. 新方法在众核平台上的性能测试

由于新广义有限元方法，无网格方法，以及粒子方法等一类新兴计算方法（外加有限差分方法）具有在不改变网格的前提下便可获得不同精度的共性特点（其总体矩阵均具有图 12(a)-(d)的变化特征）。我们相信，前述实验观察到的新广义有限元方法在众核处理器上的计算特性极有可能具有一定的普适性。

6 讨论

依据 PC 时代的经验，我们总是假设浮点运算能力是宝贵的，所以习惯于以频繁访存来避免重复计算，算法设计和优化的主要原则也是减少浮点运算次数。但是，从目前的技术发展趋势看，未来科学与工程应用所使用的计算平台的硬件计算能力与访存能力的比率必将越来越高。因此从硬件角度看，设计算法时单位访存所进行的浮点运算越多越好。这与过去科学与工程应用算法设计的思考方向和编程习惯正好相反。对于领域应用专家而言，有必要结合硬件的特征和变化趋势，对关键数值算法进行重新审视、选优，甚至重新设计；在传统的精度、收敛性等指标之外，重视计算密集度、数据移动复杂性等算法特征。

本文抛砖引玉，面向新兴计算机体系结构，从充分释放众核处理器"冗余"计算能力的角度出发，探索了针对下一代处理器结构发展高效数值算法的一个可能的技术思路。

致谢：本项目得到中科院"百人计划"项目、国家自然科学基金项目（项目编号：91130026，11472274）的资助。

参考文献：

[1] DOE Office of Science Summary report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. The opportunities and challenges of exascale computing, Fall 2010

[2] http://nvworld.ru/files/articles/calculations-on-gpu-advantagesfermi/fermipeformance.pdf

[3] Wm. A. Wulf and Sally A. McKee. Hitting the memory wall: implications of the obvious. Computer Architecture News 1995; 23(1):20-24.

[4] http://www.ucompower.com/tag/%E5%86%85%E5%AD%98%E5%A2%99/ "突破"内存墙"，CPU 可以性能更好，能耗更低"

[5] http://server.51cto.com/Fitting-171928.htm "内存墙面前 多核处理器的哭泣"

[6] Tian R(2013) Meshfree/GFEM in hardware-efficiency prospective. Interaction and multiscale mechanics. DOI:10.12989/imm. 2013.6.2.000.

[7] Tian R(2014) Simulation at Extreme-Scale: Co-Design Thinking and Practices. Arch Computat Methods Eng DOI 10.1007/s11831-014-9095-y.

[8] http://www.exascale.org/iesp/Main_Page

[9] http://www.cstam.org.cn/templates/lxxh_1/index.aspx?nodeid=65&page=ContentPage&contentid=172572 第三届面向百亿亿级计算的协同设计国际研讨会

[10] 田荣, 孙凝晖(2013) 关于我国百亿亿级计算发展的思考.《中国计算机学会通讯》 9(2): 40-48

[11] 田荣(2012) 面向百亿亿级计算协同设计的思考.《信息技术快报》 70(3): 50-63

[12] 田荣等译(2012) 百亿亿级计算机遇与挑战.《信息技术快报》, 2012; 70(3): 1-49 (原文: DOE Office of Science Summary report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. The opportunities and challenges of exascale computing, Fall 2010)

[13] Tian R(2011) Petascale Simulation and Codesign-Thinking towards Exascale Computing. The Fourth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP 2011), Dec 9-11, 2011, Tianjin, China

[14] Tian R(2013) Co-designing numerical algorithms with emerging architectures. Advances in Computational Mechanics A Conference Celebrating the 70th B-day of TJR Hughes, Feb 24-27, 2013, San Diego, USA

[15] Thibodeau P (2012) Exascale unlikely before 2020 due to budget woes. Computer world. Nov 19, 2012

[16] Harrod W (2012) DOE exascale computing Initiative (ECI) update. DOE, Office of Science (SC), Oct 4, 2012

[17] Dongarra J (2013) Emerging heterogeneous technologies for high performance computing. 22nd International Heterogeneity in Computing Workshop. IPDP, Boston

[18] DOEE3 Report, http://www.er.doe.gov/ascr/ProgramDocuments/ProgDocs.html

[19] Shalf J, Dosanjh S, Morrison J (2011) Exascale computing technology challenges. VECPAR 2010. LNCS 6449:1–25

[20] Babuška I, Melenk JM. Partition of unity method. *International Journal for Numerical Methods in Engineering* 1997; 40:727 –758.

[21] Melenk JM, Babuška I. The partition of unity finite element method: basic theory and applications. *Computer Methods in Applied Mechanics and Engineering* 1996; 139:289 –314.

[22] I. Babuška, G. Caloz, and J.E. Osborn. Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM J Numerical Analysis* 1994; 31:945–981.

[23] Duarte CA, Oden JT. An h-p adaptive method using clouds. *Computer Methods in Applied Mechanics and Engineering* 1996; 139(1-4): 237-262.

[24] Oden JT, Duarte CA, Zienkiewicz OC. A new cloud-based *hp* finite element method. *Computer Methods in Applied Mechanics and Engineering* 1998; 153(1-2): 117-126.

[25] C.A. Duarte, I. Babuska, and J.T. Oden. Generalized finite element methods for three-dimensional structural mechanics problems. *Computers & Structures* 2000; 77: 215–232.

[26] C.A. Duarte, O.N. Hamzeh, T.J. Liszka, and W.W. Tworzydlo. A generalized finite element method

for the simulation of three-dimensional dynamic crack propagation. *Computer Methods in Applied Mechanics and Engineering* 2001; 190: 2227–2262.

[27] A. Simone, C.A. Duarte, E. Van der Giessen. A generalized finite element method for polycrystals with discontinuous grain boundaries. *International Journal for Numerical Methods in Engineering* 2006; 67: 1122–1145.

[28] CA Duarte and D.J. Kim. Analysis and applications of a generalized finite element method with global-local enrichment functions. *Computer Methods in Applied Mechanics and Engineering* 2008; 197(6-8): 487–504.

[29] P. O'Hara, C.A. Duarte, T. Eason. Generalized finite element analysis for three dimensional problems exhibiting sharp thermal gradients. *Computer Methods in Applied Mechanics and Engineering* 2009; 198: 1857–1871.

[30] Strouboulis T, Babuška I, Copps K. The design and analysis of the generalized finite element method. *Computer Methods in Applied Mechanics and Engineering* 2000; 181(1–3):43–69.

[31] T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method: an example of its implementation and illustration of its performance. *International Journal for Numerical Methods in Engineering* 2000; 47:1401–1417.

[32] Strouboulis T, Copps K, Babuška I. The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering* 2001; 190(32–33):4081– 4193.

[33] T. Strouboulis, L. Zhang, and I. Babuška. Generalized finite element method using mesh-based handbooks: application to problems in domains with many voids. *Computer Methods in Applied Mechanics and Engineering* 2003; 192:3109–3161.

[34] T. Strouboulis, L. Zhang, and I. Babuška. p-version of the generalized FEM using mesh-based handbooks with applications to multiscale problems. *International Journal for Numerical Methods in Engineering* 2004; 60:1639–1672.

[35] T. Strouboulis, L. Zhang, D. Wang, and I. Babuška. A posteriori error estimation for generalized finite element methods. *Computer Methods in Applied Mechanics and Engineering* 2006; 195:852–879.

[36] T. Strouboulis, I. Babuška, and R. Hidajat. The generalized finite element method for Helmholtz equation: theory, computation, and open problems. *Computer Methods in Applied Mechanics and Engineering* 2006; 195:4711–4731.

[37] T. Strouboulis, R. Hidajat, and I. Babuška. The generalized finite element method for Helmholtz equation, part II: effect of choice of handbook functions, error due to absorbing boundary conditions and its assessment. *Computer Methods in Applied Mechanics and Engineering* 2008; 197:364–380.

[38] T. Belytschko, T. Black, Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering* 1999; 45: 601–620.

[39] Shi GH. Manifold method of material analysis. *Transactions of the 9th Army Conference on Applied Mathematics and Computing*, Report No. 92-1, U.S. Army Research Office, 1991.

[40] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 1996; 139:3–47.

[41] Lancaster P, Salkauskas K. Surfaces generated by moving least squares methods. *Mathematics of Computation* 1981; 37:141–158.

[42] Powell MJD. The theory of radial basis function approximation in 1990. In Advances in Numerical Analysis, Light FW (ed.), Oxford: Clarendon Press, 1992; 105–203.

[43] Beatson RK, Light WA, Billings S. Fast solution of the radial basis function interpolation equations: domain decomposition methods. *SIAM Journal on Scientific Computing* 2000; 22:1717–1740.

[44] 张雄, 刘岩著. 无网格法. 清华大学出版社/Springer 出版社. 2004 年 8 月第 1 版.

[45] 刘欣著. 无网格方法. 科学出版社. 2011

[46] Tian R, Extra-dof-free and linearly independent enrichments in GFEM. Comput Method Appl Mech Eng, 第 266 卷, 1–22 页, 2013

[47] Tian R, Wen Longfei, Improved XFEM (*i*XFEM)--an extra-dof free, well-conditioning, and interpolating XFEM. Comput Method Appl Mech Eng, 2014(接收待刊出)

[48] Wen Longfei, Tian R, *i*XFEM--an extra-dof free, well-conditioning, and interpolating XFEM. Proceeding of International Conference on Computational Methods, Cambridge, UK, July 28-30, 2014 (best paper award).

作者简介：

田荣： 中国科学院计算技术研究所 研究员 rtian.ict@gmail.com